AMENDMENTS TO THE CLAIMS

Please amend the claims as follows:

1. (Currently amended) A method comprising:

receiving an access request for a program object;

performing a combined check for a null reference and for a read barrier for the program object, the combined check including performing a speculative load in response to the read request and performing a speculation check for the speculative load; and

if the ~~combined check is affirmative~~ speculative load is unsuccessful, performing a recovery operation including:

determining if a read barrier is needed, and

if a read barrier is needed, performing a read barrier process.

2. (Currently amended) The method of claim 1, wherein ~~performing the combined check comprises:~~ determining if the read barrier is needed includes determining whether a least significant bit of an address of the program object is set.

~~performing a speculative load in response to the read request; and~~

~~determining whether the speculative load is successful~~.

3. (Original) The method of claim 1, wherein the method is performed in a managed runtime environment (MRTE).

4. (Original) The method of claim 1, wherein the access request is a single-byte access and further comprising implementing the access request as a multiple-byte access by reading one or more preceding or succeeding bytes of data.

5. (Current amended) A method comprising:

receiving a subject code, the subject code including an access to a program object ~~that may be guarded~~; and

compiling the subject code into machine executable code, the machine executable code including a read barrier check for the program object, the read barrier check comprising:

performing a speculative load to access the program object;

performing a speculation check for the speculative load; and

if the speculative load of the access fails, performing a recovery including:

determining if a read barrier is needed;

if a read barrier is needed, performing a read barrier process.

6. (Original) The method of claim 5, wherein the speculative load operates as a combined check of a null reference and a check for the read barrier.

7. (Original) The method of claim 5, wherein the object code is compiled for a managed runtime environment (MRTE).

8. (Original) The method of claim 5, wherein the recovery comprises testing whether the failure of the speculative load results from an access to a program object that is guarded.

9. (Original) The method of claim 8, wherein testing whether the failure of the speculative load results from an access to a program object that is guarded comprises determining whether a bit for the address of the program object is set.

10. (Original) The method of claim 5, wherein the recovery comprises testing whether the failure of the speculative load results from a null reference.

11. (Currently amended) A system comprising;

a processor; and

a compiler to be run by the processor, the compiler to:

receive subject code, the subject code including an access to a program object ~~that may be guarded~~; and

compile the subject code into machine executable code, the machine executable code including a read barrier check for the program object, the read barrier check comprising:

performing a speculative load to access to program object;

performing a speculation check for the speculative load; and

if the speculative load of the access fails, performing a recovery including:

determining if a read barrier is needed;

if a read barrier is needed, performing a read barrier process.

12. (Original) The system of claim 11, wherein the speculative load operates as a combined check of a null reference and a check for the read barrier.

13. (Original) The system of claim 11, wherein compiling the object code comprises providing for setting a bit when a program object is guarded.

14. (Original) The system of claim 11, wherein the system comprises a managed runtime environment (MRTE).

15. (Original) The system of claim 11, wherein the recovery comprises testing whether the failure of the speculative load results from an access to a program object that is guarded.

16. (Original) The system of claim 15, wherein testing whether the failure of the speculative load results from an access to a program object that is guarded comprises determining whether a bit for the program object is set.

17. (Original) The system of claim 16, wherein the recovery comprises testing whether the failure of the speculative load results from a null reference.

18. (Currently amended) A ~~machine-readable~~ computer-readable medium having stored thereon data representing sequences of instructions that, when executed by a processor, cause the processor to perform operations comprising:

receiving an access request for a program object;

performing a combined check for a null reference and for a read barrier for the program object, the combined check including performing a speculative load in response to the read request and performing a speculation check for the speculative load; and if the combined check is affirmative, performing a recovery operation, the

recovery operation including:

determining if a read barrier is needed;

if a read barrier is needed, performing a read barrier process.

19. (Currently amended) The medium of claim 18, wherein ~~performing the combined check comprises:~~ determining if the read barrier is needed includes determining whether a least significant bit of an address of the program object is set.

~~performing a speculative load in response to the read request; and~~

~~determining whether the speculative load is successful.~~

20. (Original) The medium of claim 18, wherein the method is provided in a managed runtime environment (MRTE).

21. (Original) The medium of claim 18, wherein the access request is a single-byte access and wherein the instructions further comprise instructions comprising implementing the access request as a multiple-byte access by reading one or more preceding or succeeding bytes of data.

22. (Currently amended) A ~~machine-readable~~ computer-readable medium having stored thereon data representing sequences of instructions that, when executed by a processor, cause the processor to perform operations comprising:

receiving subject code, the subject code including an access to a program object

~~that may be guarded~~; and

compiling the subject code into machine executable code, the machine executable code including a read barrier check for the program object, the read barrier check comprising:

performing a speculative load to access the program object;

performing a speculation check of the speculative load; and

if the speculative load of the access fails, performing a recovery <u>including:</u>

<u>determining if a read barrier is needed;</u>

<u>if a read barrier is needed, performing a read barrier process.</u>

23. (Original) The medium of claim 22, wherein the speculative load operates as a combined check of a null reference and a check for the read barrier.

24. (Original) The medium of claim 22, wherein compiling the subject code comprises providing for setting a bit when a program object is guarded.

25. (Original) The medium of claim 22, wherein the subject code is compiled for a managed runtime environment (MRTE).

26. (Original) The medium of claim 22, wherein the recovery comprises testing whether the failure of the speculative load results from an access to a program object that is guarded.

27. (Original) The medium of claim 27, wherein testing whether the failure of the speculative load results from an access to a program object that is guarded comprises determining whether a bit for the program object is set.

28. (Original) The medium of claim 22, wherein the recovery comprises testing whether the failure of the speculative load results from a null reference.